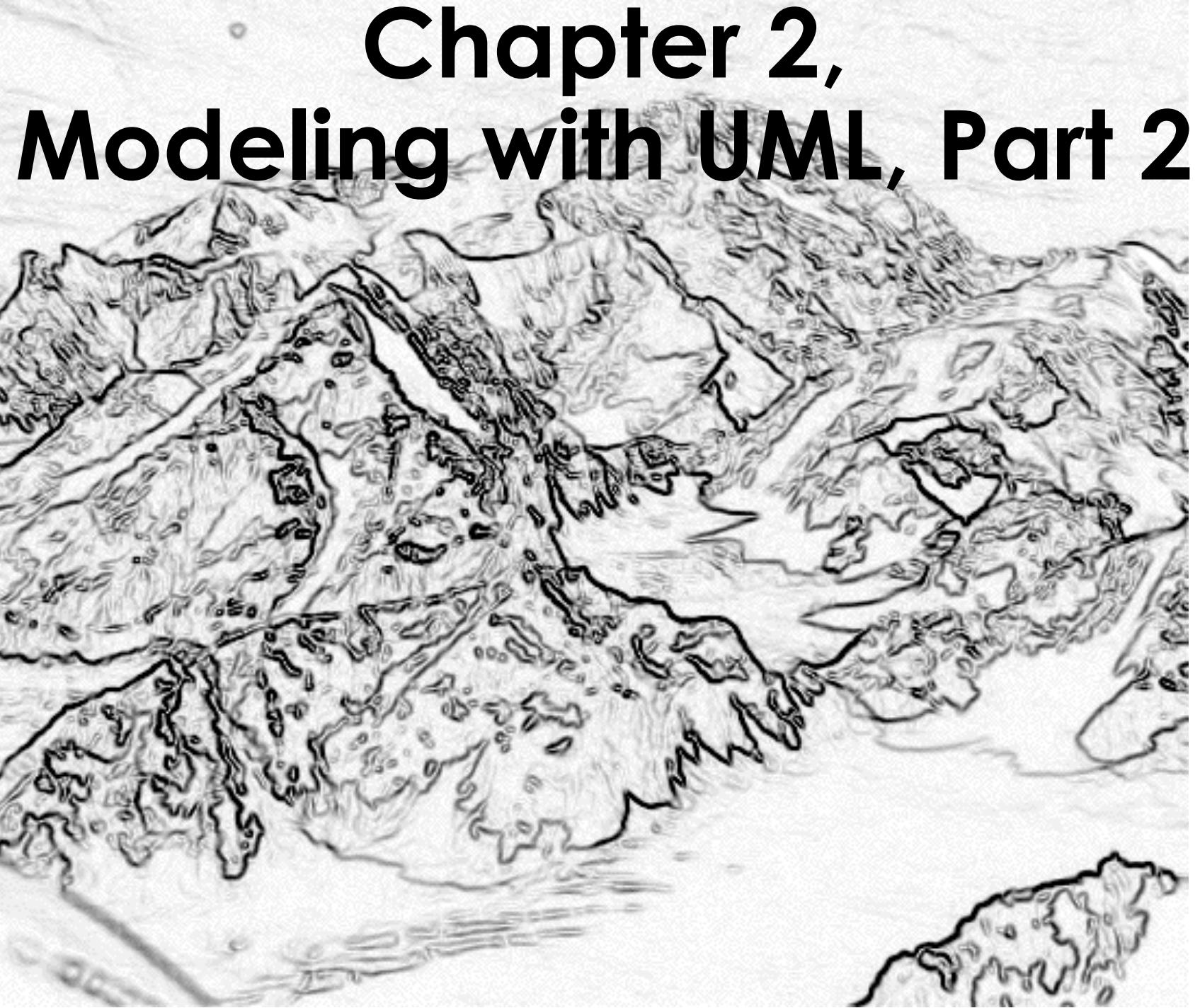


Object-Oriented Software Engineering

Using UML, Patterns, and Java



Chapter 2, Modeling with UML, Part 2

Unified Modeling Language

- un linguaggio (e notazione) universale, per la creazione di modelli software
- nel novembre '97 è diventato uno **standard** approvato dall'**OMG** (Object Management Group)
- numerosi i co-proponenti: Microsoft, IBM, Oracle, HP, Platinum, Sterling, Unysis

What is UML? Unified Modeling Language

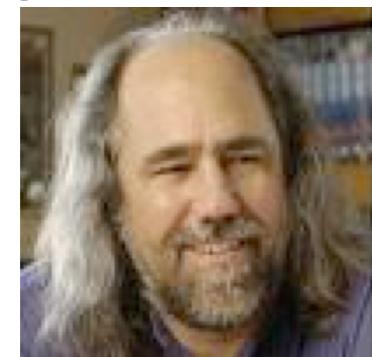
- Convergence of different notations used in object-oriented methods, mainly
 - OMT (James Rumbaugh and colleagues), OOSE (Ivar Jacobson), Booch (Grady Booch)
- They also developed the Rational Unified Process, which became the Unified Process in 1999



25 years at GE Research, where he developed OMT, joined (IBM) Rational in 1994, CASE tool OMTool



At Ericsson until 1994, developed use cases and the CASE tool Objectory, at IBM Rational since 1995,
<http://www.ivarjacobson.com>



Developed the Booch method (“clouds”), ACM Fellow 1995, and IBM Fellow 2003
<http://www.booch.com/>

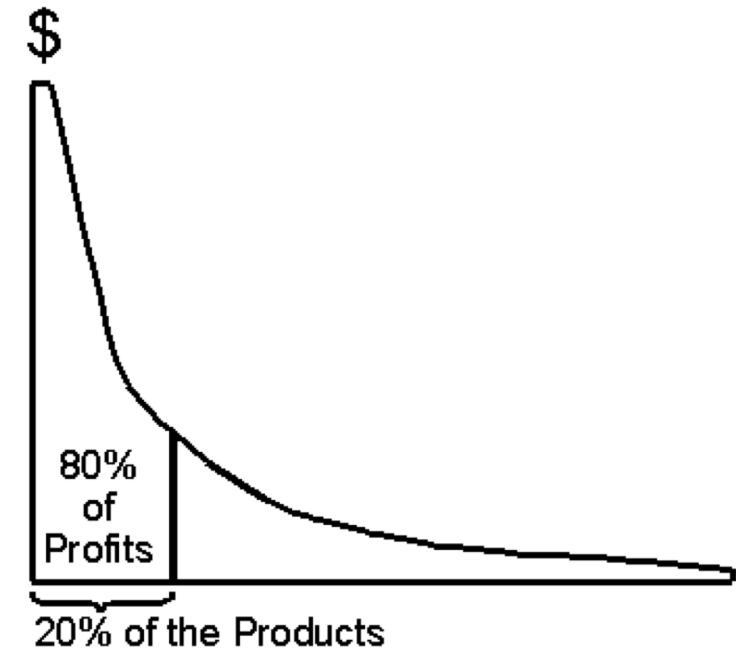
UML

- Nonproprietary standard for modeling systems
- Current Version: UML 2.5
 - Information at the OMG portal <http://www.uml.org/>
- Commercial tools:
 - Rational (IBM), Together (Borland), Visual Architect (Visual Paradigm), Enterprise Architect (Sparx Systems)
- Open Source tools <http://www.sourceforge.net/>
 - ArgoUML, StarUML, Umbrello (for KDE), PoseidonUML
- Example of research tools: Unicase, Sysiphus
 - Based on a unified project model for modeling, collaboration and project organization
 - <http://unicase.org>
 - <http://sysiphus.in.tum.de/>



UML: First Pass

- You can solve 80% of the modeling problems by using 20 % UML
- We teach you those 20%
- 80-20 rule: Pareto principle



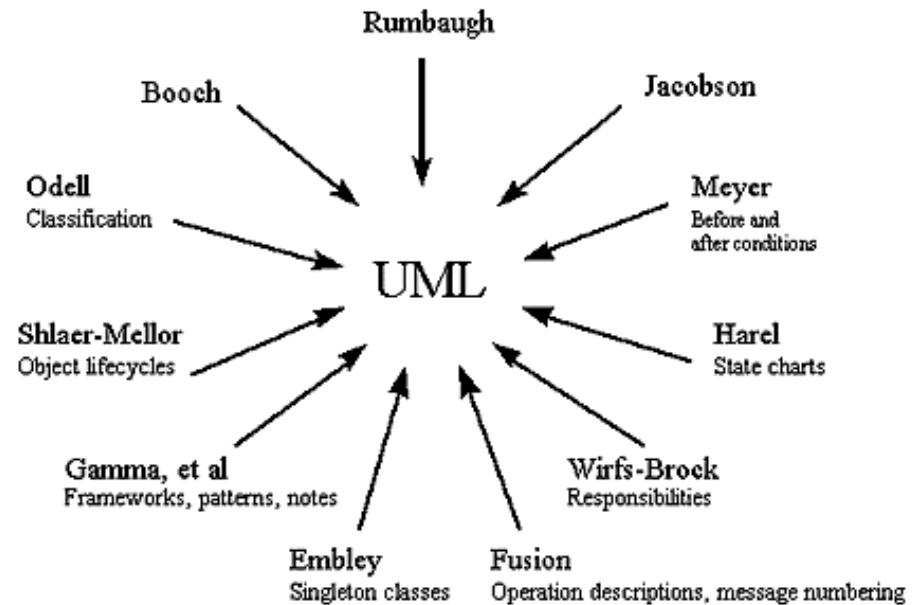
Vilfredo Pareto, 1848-1923
Introduced the concept of Pareto Efficiency,

Founder of the field of microeconomics.

Unified Modeling Language

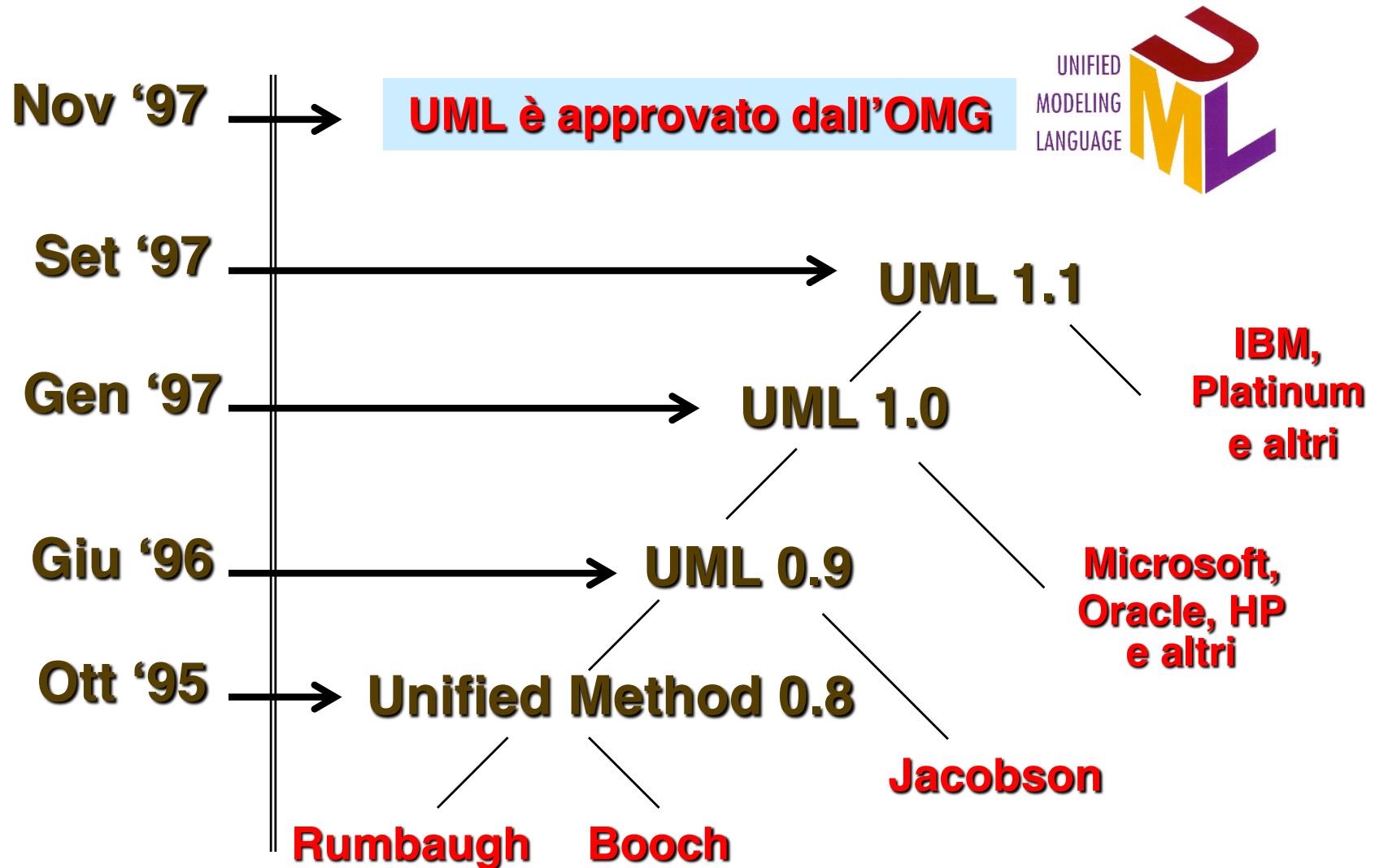
- è l'unificazione dei metodi:

- Booch-93 di **Grady Booch**
- OMT di **Jim Rumbaugh**
- OOSE di **Ivar Jacobson**

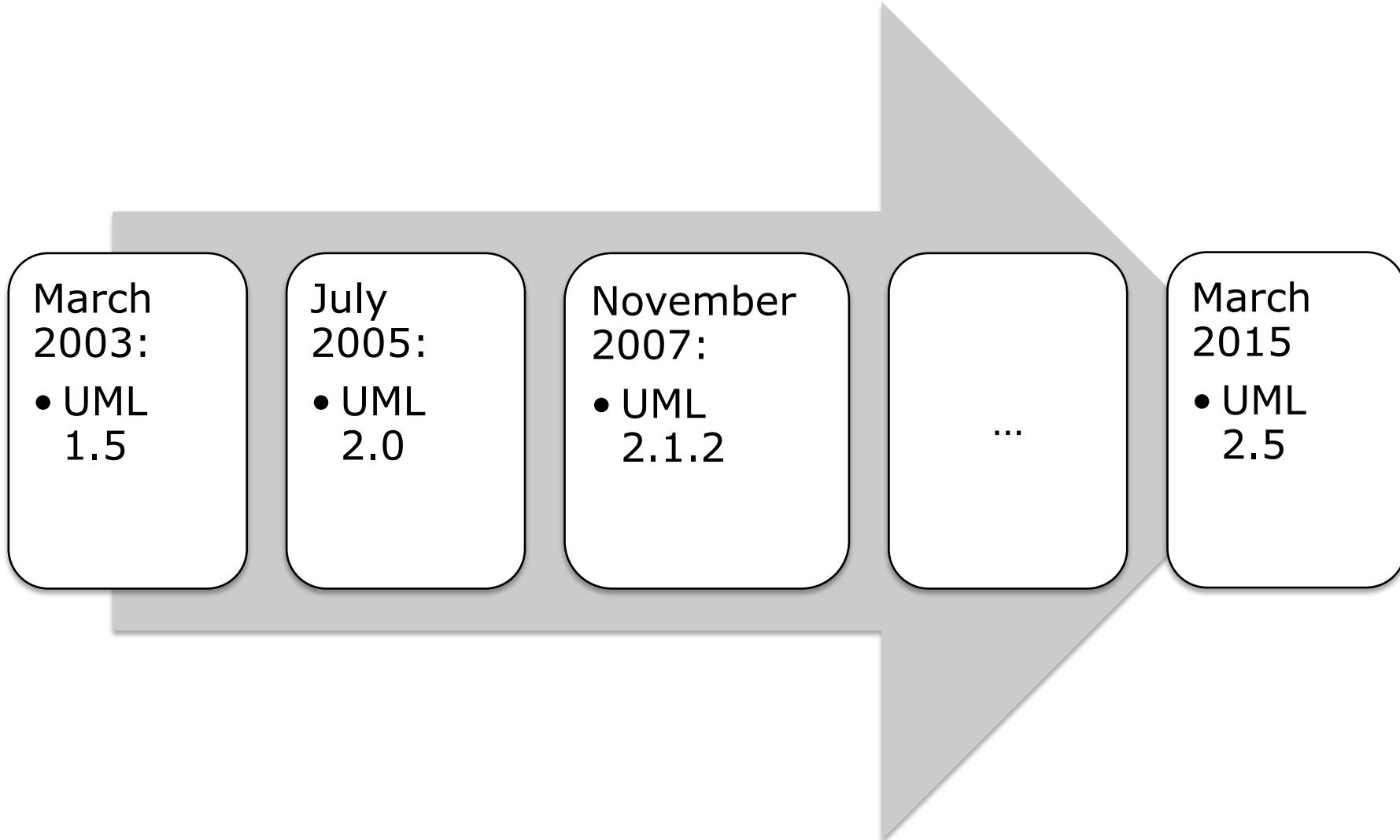


- ha accolto inoltre le idee di numerosi altri metodologi
- è tuttavia indipendente dai metodi, dalle tecnologie, dai produttori

Storia di UML...



Recent History of UML



UML: recent versions

- UML now usable with Model Driven Architecture (MDA)
 - Better support for the automatic transformation of a Platform Independent Model (PIM) into a Platform Specific Model (PSM)
- UML 2 is based on a meta model
 - Meta Object Facility (MOF)

UML - linguaggio universale

- **linguaggio** per specificare, costruire, visualizzare e documentare gli artefatti di un sistema
- **universale**: può rappresentare sistemi molto diversi, da quelli web ai legacy, dalle tradizionali applicazioni Cobol a quelle object oriented e a componenti

UML non è un metodo

- è un **linguaggio di modellazione**, non un metodo, né una metodologia
- definisce una notazione standard, basata su un metamodello integrato degli “elementi” che compongono un sistema software
- non prescrive una sequenza di processo, cioè non dice “prima bisogna fare questa attività, poi quest’altra”

UML e Processo Software

*“un unico processo universale buono per tutti gli stili
dello sviluppo non sembra possibile e tanto meno
desiderabile”*

- in realtà UML assume un processo:
 - basato sui Casi d'Uso (**use case driven**)
 - incentrato sull'architettura
 - iterativo e incrementale
- i dettagli di questo processo di tipo generale vanno adattati alle peculiarità della cultura dello sviluppo o del dominio applicativo di ciascuna organizzazione

UML: meta-modello

- UML si fonda su un meta-modello integrato, che definisce le caratteristiche e le relazioni esistenti tra i diversi elementi (classi, attributi, moduli, ...)
- Il meta-modello è la base per l'implementazione dell'UML da parte dei produttori di strumenti di sviluppo (CASE, ambienti visuali, ...) e per l'interoperabilità tra i diversi strumenti

UML: meta-modello

- UML prevede una serie di **modelli diagrammatici** basati sul meta-modello
- gli elementi del meta-modello possono comparire in diagrammi di diverso tipo
- alcuni elementi (ad es. la “classe”) hanno una icona che li rappresenta graficamente

Architettura a quattro livelli

- UML è basato su un modello architettonico a quattro livelli

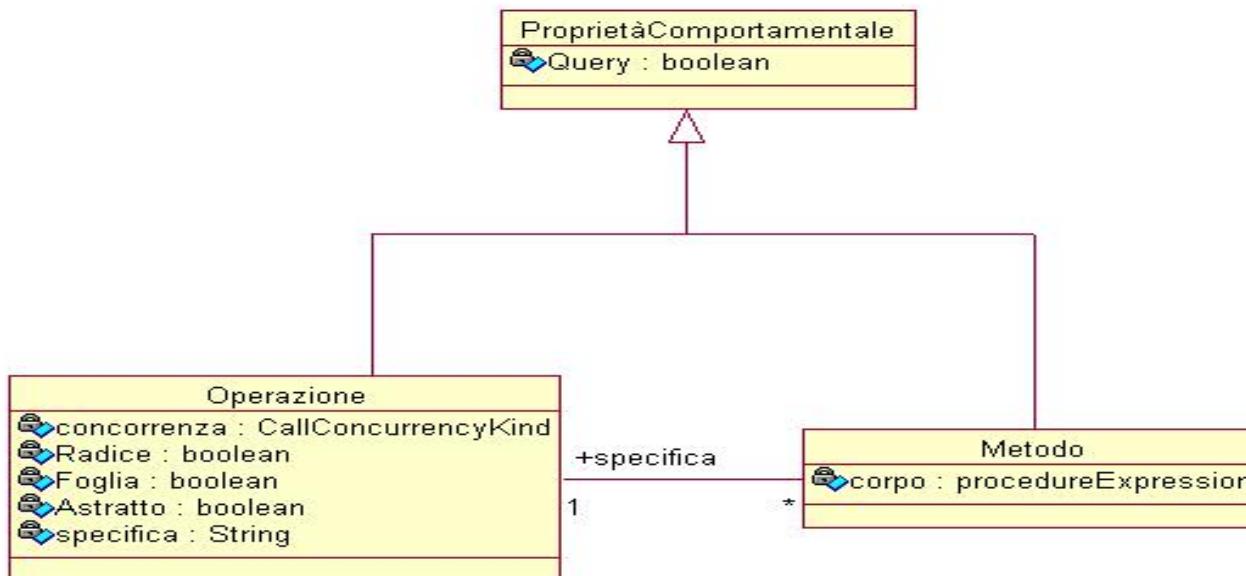
Livello	Descrizione	Esempio
Meta-meta-modello	Definisce un linguaggio per la specifica dei metamodelli	Meta-classi, meta-attributi, meta-operazioni, ...
Meta-modello	Definisce un linguaggio per la specifica dei modelli	classi, attributi, operazioni, ...
Modello	Definisce un linguaggio per la specifica di un certo dominio applicativo	Classi di uno specifico contesto
Oggetti	Istanze del modello	Oggetti ottenuti istanziando le classi del modello

Sintassi astratta

- Nell'architettura a quattro livelli di UML ogni livello è un'astrazione di quello sottostante, ed è definito in termini di quello sovrastante.
- La sintassi astratta di UML viene definita utilizzando la notazione dei metamodelli. Si tratta di un sottoinsieme della notazione di UML che utilizza il diagramma delle classi di UML per definire gli elementi dei modelli UML e le relazioni che intercorrono tra di loro.
- La sintassi astratta è espressa tramite diagrammi e linguaggio naturale.

Frammento della sintassi astratta di UML

- Questo diagramma mostra che Operazione e Metodo sono (meta)classi, entrambe sottoclassi della (meta)classe astratta proprietà comportamentale



Regole sintattiche

- Le regole sintattiche che generano espressioni ben formate si applicano ad istanze delle metaclassi (cioè a operazioni, metodi, classi e tutto ciò che si trova a livello del modello)
- Esse forniscono le regole cui devono obbedire le istanze delle metaclassi
- Sono descritte tramite Object Constraint Language (OCL) e linguaggio naturale

Semantica

- La semantica è descritta tramite il linguaggio naturale con il supporto di alcuni diagrammi
- La semantica include anche
 - La guida alla notazione
 - La gestione dei modelli

Obiettivi dell'UML

- Fornire all'utente un linguaggio di specifica espressivo, visuale e pronto all'uso
- Offrire meccanismi di estensibilità e specializzazione del linguaggio
- Essere indipendente dagli specifici linguaggi di programmazione e dai processi di sviluppo
- Incoraggiare la crescita dei tool OO commerciali
- Supportare concetti di sviluppo ad alto livello come frameworks, pattern ed i componenti
- Integrare i migliori approcci.

Cosa non è UML!

- Non è un linguaggio di programmazione visuale
(è un linguaggio di specifica visuale)
- UML non è un modello per la definizione di interfacce
- UML non è dipendente dal paradigma di sviluppo nel quale può essere utilizzato

We use Models to describe Software Systems

- **System model:** functional model + object model + dynamic model
- **Functional model:** What are the functions of the system?
 - UML Notation: Use case diagrams
- **Object model:** What is the structure of the system?
 - UML Notation: Class diagrams
- **Dynamic model:** How does the system react to external events?
 - UML Notation: Sequence, State chart and Activity diagrams

Diagrams

- **Use case diagrams**
 - Describe the functional behavior of the system as seen by the user
- **Class diagrams**
 - Describe the static structure of the system: Objects, attributes, associations
- **Sequence diagrams**
 - Describe the dynamic behavior between objects of the system
- **Statechart diagrams**
 - Describe the dynamic behavior of an individual object
- **Activity diagrams**
 - Describe the dynamic behavior of a system, in particular the workflow.

- **Packages**
 - Packages help you to organize UML models to increase their readability
 - We can use the UML package mechanism to organize classes into subsystems
- **Component diagrams**
 - Definiscono le relazioni fra i componenti software che realizzano l'applicazione sorgenti, binari, eseguibili, ...
 - Evidenziano l'organizzazione e le dipendenze esistenti tra componenti
- **Deployment diagrams**
 - Permette di rappresentare, a diversi livelli di dettaglio, l'architettura fisica del sistema - la configurazione dei nodi elaborativi in ambiente di esecuzione (run-time)

Diagrams will be explained in details in the next lessons